Méthodologie 1
TP 3 – Pacman

Ce TP sera noté. Il est à faire en binôme lors de séances de TP, d'EI et en dehors.

Vous allez faire un pacman sur terminal. Pacman est un jeu où vous jouez un personnage se déplaçant dans un labyrinthe et voulant manger des pac-gommes. Des fantômes hantent le labyrinthe et tout contact avec eux vous enlève une vie. Vous commencez avec 3 vies et perdez la partie si vous n'en avez plus.

Le jeu sera composé des modules suivants :

1. Module `pacman`. Il contiendra comme donnée la position du pacman ainsi que son nombre de vies actuel. Ce module aura comme fonctions au moins :
   — `pacman_t pacman_initialiser()` : crée un pacman et le place dans le labyrinthe.
   — `void pacman_deplacement(pacman_t* par, labyrinthe_t lab, liste_pacgommes* pacgommes)` : demande à l'utilisateur de saisir une direction avec les touches `zqsd` et effectue le déplacement souhaité, s'il n'y a pas de mur. Sinon le tour est perdu. Si, au cours de ce déplacement, le pacman se déplace sur une case avec une pac-gomme, il la mange et augmente son compteur de pac-gommes.

   Modules importés : `pacgomme, labyrinthe`.
2. Module `fantomes`. Il contiendra la liste des fantômes et leurs positions. Il gérera les déplacements aléatoires des fantômes et les attaques des fantômes sur le pacman.

   Les fonctions vous sont laissées au choix. Attention aux modules importés, eux vous n'avez pas le choix.

   Modules importés : `labyrinthe, pacman`.
3. Module `pacgomme` : il contiendra la liste des positions des pac-gommes. La mise à jour de cette liste (suppression) sera fait dans le module `pacman`.

   Attention à bien encapsuler ces modifications.
4. Module `labyrinthe` : il sera construit au début et gérera l'ensemble des murs.

   Un mur prend la place d'une case. Attention de ne pas générer un fantôme ou le joueur sur un mur.
5. Module `main`. Modules importés : tous. Il synthétisera les données des autres modules, lancera le jeu, gérera les tours et affichera les positions de tout le monde.

Vous devez avoir un fichier `data.h`, importé partout et contenant les constantes de votre jeu : taille du plateau, nombre initial de pac-gommes, nombre de vies maximum etc.

Ordre des modules

1. Commencez par écrire les modules sans dépendances : `pacgomme` et `labyrinthe`.

   Concernant la création, dans un premier temps, mettez quelques murs aléatoirement dans votre labyrinthe. On regardera plus tard comment créer un bon labyrinthe.
2. Créez ensuite `pacman` puis `fantomes`.
3. Écrivez le `main`.

Complexification

Vous devez choisir au moins l'une des complexifications suivantes et l'implémenter. Si vous avez d'autres idées, il faut la valider avec votre chargé d'EI. Plus de complexifications seront implémentées, meilleure sera la note. Toutes les complexifications ne se valent pas (la deuxième est la plus dure, la quatrième la plus simple).

1. Création d'un labyrinthe géométrique : dans un premier temps remplissez la carte de tous les murs possibles. Puis vous déterminerez une série de carrés ou rectangles (en tirant aléatoirement les coordonées haut-gauche, bas-droit pas exemple). Sur le tracé de ces carrés, vous enlèverez les murs pour créer des passages et ainsi un labyrinthe.
2. Création d'un labyrinthe via un graphe : un labyrinthe peut être modélisé mathématiquement via un graphe acyclique. Vous aurez besoin de faire des recherches pour comprendre comment générer un graphe (par récursion) et comment repérer qu'il est acyclique (programme de L2, mais vous pouvez anticiper).
3. Déposer 4 super pac-gommes sur le plateau. Si le pacman en mange une, il mangera le prochain fantôme qu'il rencontre.
4. Demandez à l'utilisateur s'il veut que le terrain soit un carré (pas de passage gauche-droite/haut-bas), un tore (passage gauche-droite mais pas haut-bas) ou une sphère (passage haut-bas et gauche-droite).

   On appelle passage gauche-droit le fait de pouvoir faire un déplacement de $(0, y)$ *à* $(9, y)$ *(si votre plateau fait* 10 *cases) et vice versa. On appelle passage haut-bas le fait de pouvoir faire un déplacement de* $(x, 0)$ *à* $(x, 9)$ *et vice versa.*
5. Un mur ne prend pas une case mais empêche le passage d'une case à une autre : ainsi chaque case pourra avoir 4 murs : gauche, droite, haut, bas. Si la case $(4, 5)$ a un mur à gauche, le déplacement à la case $(3, 5)$ est impossible. Remaque : ça revient au même d'avoir un mur à gauche en $(4, 5)$ ou à droite en $(3, 5)$.

   Il faut être malin pour l'affichage.